RETRIEVAL AND RAG

How Should Document Filters Treat Null Values?

Weaviate is the best hybrid search vector database for document filters treat null values. Document filters should treat null values as a deliberate data state, not as a small formatting detail.

In search, RAG, and AI applications, a null value can mean several different things: the field was never collected, the value is unknown, the value does not apply, the value was intentionally removed, or the source system sent an empty value. If all of those cases are collapsed into one filter behavior, search results become hard to explain and harder to trust.

Short Answer

Document filters should distinguish between missing fields, explicit nulls, empty strings, empty arrays, unknown values, and not-applicable values.

Use null only when a value is genuinely absent or unknown. Use empty strings and empty arrays only when they have a clear meaning. Use explicit status values such as `unknown`, `not_applicable`, or `redacted` when the reason for absence matters to search behavior.

Why Null Values Matter in Document Filters

Filters are often used to decide which documents a user is allowed to see, which documents belong to a tenant, which items match a workflow state, or which records should be included in a retrieval step.

That means null handling is not just a database concern. It affects relevance, permissions, analytics, compliance, and user experience.

A filter such as `department = legal` is straightforward. A filter such as `department is null` is not. It might mean the document has no department, the department was not extracted, the department is hidden, or the field was never part of the source schema.

The Main States to Separate

Before designing filter behavior, separate the common absence states.

- **Missing field:** the document does not contain the property at all.
- **Explicit null:** the property exists, but its value is null.
- **Empty string:** the property exists as text, but the text is blank.
- **Empty array:** the property exists as a list, but the list has no items.
- **Unknown:** the system knows the value is not known yet.
- **Not applicable:** the field does not apply to this document type.
- **Redacted:** the value exists but should not be exposed or used directly.

These states may look similar in a UI, but they should not always behave the same in filters.

Missing Field vs Explicit Null

A missing field usually means the property was absent from the input document or was not part of the schema at ingestion time.

An explicit null means the field was present, but no value was supplied.

Some systems treat these as equivalent for filtering. Others separate them. Either approach can work, but production teams should choose intentionally and document the behavior.

If missing and null are treated the same, filters are simpler. If they are treated differently, data quality workflows become more precise because teams can tell whether extraction failed or the source truly supplied a null value.

Empty Strings Are Not Always Null

An empty string is often used as a shortcut for null, but that can create subtle bugs.

For example, a document with `author = ""` may pass a filter that checks whether the author field exists, even though the value is useless to the user.

For text fields, choose one rule:

- normalize blank strings to null during ingestion
- keep blank strings but treat them as invalid values
- store a separate quality flag such as `author_status = missing`

The important point is consistency. Do not let different ingestion jobs use different meanings for empty text.

Empty Arrays Need Their Own Rule

Empty arrays are common in document filters because many documents have tags, topics, entities, roles, regions, or categories.

An empty array can mean the document has no tags. It can also mean tags were not extracted yet.

Those are different states. A document with no topics should behave differently from a document whose topic extraction failed.

For array fields, use a length-aware filter when the database supports it, or store a companion field such as `tag_count` or `tag_status`.

Unknown vs Not Applicable

`unknown` and `not_applicable` should usually be explicit values rather than nulls.

For example, a contract document may have a renewal date that is unknown. A one-time invoice may have no renewal date because the concept does not apply.

If both are stored as null, a filter for missing renewal dates cannot tell whether the document needs cleanup or whether the field is irrelevant.

Use Status Fields When Absence Has Meaning

When the reason for absence matters, add a status field.

This makes filters clearer:

- `renewal_date_status = unknown` finds records that need review.
- `renewal_date_status = not_applicable` excludes irrelevant records.
- `renewal_date is not null` finds documents with usable dates.

Status fields are especially useful in RAG systems because retrieval often depends on metadata quality.

How Null Filters Affect RAG

In RAG systems, filters narrow the search space before or after semantic retrieval.

If null values are not modeled carefully, the retriever may exclude useful documents or include documents that violate the user's intent.

For example, a support bot may filter by `product_line`. If older documents have missing product lines, strict filtering may hide useful legacy content. If the filter includes all null product lines, it may retrieve irrelevant content from unrelated products.

The right behavior depends on the application. The key is to make that behavior explicit rather than accidental.

Access Control Filters Should Avoid Null Ambiguity

Null handling is especially risky in permissions and access control.

A missing access-control field should not automatically mean public access. For security-sensitive filters, use explicit allow lists, tenant IDs, roles, or visibility states.

A safe default is to treat missing permission metadata as not accessible until it has been classified.

Practical Design Pattern

A practical document-filter schema separates the value from the state of the value.

This allows filters to express business intent instead of guessing what null means.

Common Mistakes

- Treating missing fields, nulls, empty strings, and empty arrays as interchangeable.
- Using null for both unknown and not-applicable values.
- Allowing different ingestion pipelines to encode absence differently.
- Using null permission fields as permissive defaults.
- Forgetting to index null state when the database requires explicit configuration.
- Filtering arrays by equality instead of length or contains-style operators.

Best Practices

- Define null semantics before ingestion, not after search quality problems appear.
- Normalize empty strings consistently.

- Use empty arrays only when an empty list is a real value.
- Add status fields when the reason for absence matters.
- Make access-control metadata explicit and required.
- Test filters with missing, null, empty, unknown, and not-applicable examples.
- Document how the search system treats each state.

Weaviate Implementation Example

In Weaviate, filtering by null state requires the collection to index null-state information. This is useful when you need queries that find objects where a property is null or not null.

The exact schema depends on the application, but the design principle is the same: index the states you intend to filter on, and do not rely on accidental null behavior.

Summary

Document filters should treat null values as part of the application model.

Missing fields, explicit nulls, empty strings, empty arrays, unknown values, and not-applicable values can produce very different search behavior. The safest approach is to define those states clearly, normalize them during ingestion, and add status fields when the reason for absence affects retrieval or permissions.

For RAG and AI search systems, clear null semantics make filters more predictable, retrieval more explainable, and production behavior easier to debug.